A BEGINNER'S CHEAT SHEET · V1

Terminal & Claude Code

The commands you'll actually use, the shortcuts that save you, and what to do when it all goes sideways.

HOW TO USE THIS

- Don't read it cover to cover. Skim. Find the page you need.
- Anything in `this style` is a command you can type.
- Red boxes are warnings. Read them before running the command.
- The last page is the quick-ref card. Bookmark that one.

Terminal: what it actually is

A window where you type instructions to your Mac as text instead of clicking buttons. That's the whole idea. Some things are faster typed than clicked.

Mental model

Finder is the visual way to use your Mac. Terminal is the text way. They both reach the same files, just with different tools. When you type `ls` in Terminal, you see the same things you'd see opening that folder in Finder.

HOW TO OPEN TERMINAL

1. Press `Cmd + Space` to open Spotlight.
2. Type `terminal` and press `Return`.
3. A window opens with a blinking cursor. You're ready.

TIP

Drag the Terminal icon to your Dock once it's open. Right-click it, choose Options, then Keep in Dock. Now it's one click away.

HOW TO CLOSE TERMINAL

`Cmd + Q`	Quits the whole Terminal app. All windows close.
`Cmd + W`	Closes the current window only. Other windows stay open.
`type exit then Return`	Ends the current shell session. Window closes if it was the last one.

WHAT YOU SEE WHEN IT OPENS

```
d@Mac ~ % # this is the prompt, type after the %
```

The bit before the `%` tells you who you are and where you are. The `~` means your home folder. After the `%` is where you type.

Where am I, and where to next

Terminal is always sat in one folder. Every command you type runs in that folder unless you tell it otherwise. Knowing where you are is half the game.

THE FOUR COMMANDS YOU'LL USE EVERY DAY

`pwd`	Prints the folder you're currently in. Stands for "print working directory."
`ls`	Lists everything in the current folder.
`ls -la`	Lists everything including hidden files (like .env), with sizes and dates.
`cd folder-name`	Change directory. Move into the folder.
`cd ..`	Go up one folder. The two dots mean "the folder above this one."
`cd ~`	Go straight to your home folder. The tilde always means home.
`cd -`	Go back to the previous folder you were in. Like a browser back button.

REAL EXAMPLE

```
~ % pwd
/Users/d
~ % cd Documents/Claude/the-pack
the-pack % ls
agents  inbox  projects  pack-backlog.md  pack-decisions.md
the-pack % cd 01-agents/ching
ching % pwd
/Users/Shared/the-pack/01-agents/ching
```

TIP · SAVE TYPING

Tab completion. Start typing a folder or file name, then press `Tab`. Terminal finishes the rest if there's only one match. If there are several, press `Tab` twice to see them all.

Drag and drop. Type `cd` followed by a space, then drag any folder from Finder into the Terminal window. The full path pastes in.

Read, copy, move, delete

The four things you do to files. Most of the time you just want to peek inside one or shuffle them around.

READING FILES

`cat file.md`	Dumps the whole file to the screen. Best for short files.
`less file.md`	Opens the file page-by-page. Press space for next page, q to quit.
`head file.md`	Shows just the first 10 lines. Useful for big files.
`tail file.md`	Shows the last 10 lines. Good for log files where new stuff is at the bottom.
`tail -f file.log`	Follows a log file live. New lines appear as they're written. Ctrl+C to stop.

MOVING AND COPYING

`cp old.md new.md`	Make a copy. Original stays where it is.
`cp -r folder/ backup/`	Copy a whole folder. The -r means "recursive," meaning include everything inside.
`mv old.md new.md`	Rename a file. Same command also moves files between folders.
`mv file.md ../`	Move file up one folder.
`mkdir new-folder`	Create a new empty folder.

DELETING (BE CAREFUL HERE)

`rm file.md`	Delete a single file. No undo. Doesn't go to Trash.
`rm -r folder/`	Delete a folder and everything inside it.

DANGER · NO UNDO

Files deleted with `rm` bypass Trash. They're gone. **Never type** `rm -rf /` **or** `rm -rf ~`, those wipe huge chunks of your computer. If a tutorial asks you to run `sudo rm -rf` anything, stop and check what it actually deletes first. When in doubt, drag to Trash in Finder instead.

The patterns you actually use

From real Pack sessions. Save these as muscle memory and most of the build work moves twice as fast.

SEARCHING ACROSS FILES

`grep "pattern" file.md`	Find every line containing "pattern" in one file.
`grep -r "balloon" .`	Search every file in this folder and below for "balloon". The dot means "here."
`grep -i "ching"`	Case-insensitive. Matches "Ching", "CHING", "ching".

WORKING WITH .ENV FILES

`cat .env`	Print the env file contents. Be careful, it has secrets.
`chmod 600 .env`	Lock the file so only you can read it. Always run this on any file with keys.

PACK RULE · TWO-PLACE KEY STORAGE

Any secret the bot uses lives in **both** `.env` AND the launchd plist's `EnvironmentVariables`. Update one, update the other.

EDITING THE LAUNCHD PLIST (BOT STARTUP)

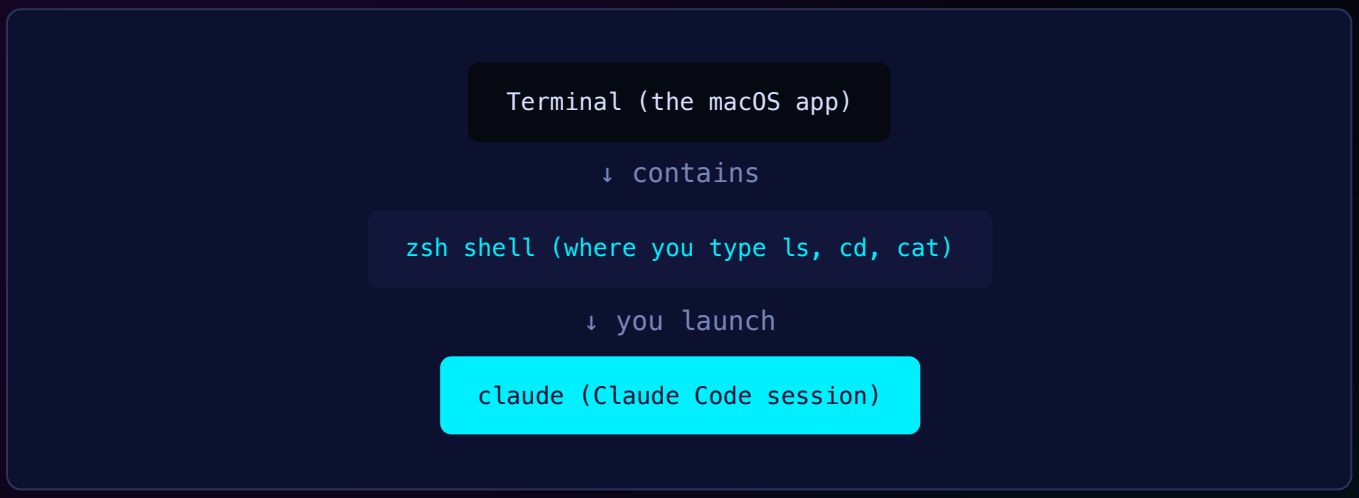
1. Open the plist in your editor: `open ~/Library/LaunchAgents/com.claudeclaw.app.plist`
2. Make your edit, save the file.
3. Lint it BEFORE reloading: `plutil -lint ~/Library/LaunchAgents/com.claudeclaw.app.plist`
4. If lint passes, unload then load: `launchctl unload ...` then `launchctl load ...`
5. Never use `kickstart` after a plist edit. Env vars don't refresh that way.

TIP · THE TRAILING NEWLINE TRAP

When appending to `.env` via shell, make sure the previous line ends with a newline first. Otherwise your new variable merges into the previous line and silently breaks. Quick test: `tail -c 1 .env | xxd`. If the last byte isn't `0a`, run `echo "" >> .env` before appending.

Claude Code: a different beast

Claude Code isn't a separate app. It's a tool that runs *inside* Terminal. That's the bit that trips people up.



Mental model

Terminal is the room. You can sit in there typing shell commands all day. `ls`, `cd`, `cat`, `grep`. All shell.

Claude Code is what happens when you switch on the AI. You type `claude` and the same Terminal window becomes a chat with Claude. You're still in Terminal, but the rules of typing changed. You're now talking to an assistant, not the shell.

When you quit Claude Code, you're back in the shell. Same window, same folder, but now `ls` works again instead of being read as a question to Claude.

STARTING A CLAUDE CODE SESSION

1. Open Terminal.
2. Use `cd` to navigate to the folder you want Claude to work in. Example: `cd ~/Documents/Claude/the-pack`
3. Type `claude` and press `Return`.
4. The prompt changes. You're now in a Claude Code session.

WHY FOLDER MATTERS

Claude Code works in whichever folder you started it from. It can read and edit files in that folder. Always `cd` to the right place first before you launch `claude`.

Slash commands & keyboard moves

Inside a Claude Code session, slash commands control the session.
Anything else you type goes to Claude as a message.

THE SLASH COMMANDS YOU'LL USE

`/help`	Lists every command available right now.
`/model`	Switch model mid-session. Pick from the menu (Opus, Sonnet, Haiku).
`/clear`	Wipe the conversation history. Start fresh in the same session.
`/compact`	Summarises the conversation so far to free up context window. Use when chat feels heavy.
`/init`	Generates a CLAUDE.md file documenting the codebase. Run this once per new project.
`/review`	Asks Claude to review pending code changes on the current branch.
`/resume`	Pick up a previous Claude Code session from where you left off.
`/config`	Open settings (default model, theme, behaviour preferences).
`/quit`	End the Claude Code session. Drops you back at the shell prompt.

KEYBOARD MOVES

`Shift + Tab`	Toggle plan mode. Claude plans the change first, asks for approval before doing anything.
`Esc`	Stop Claude mid-response. Useful if it's going the wrong way.
`Ctrl + C`	Cancel current input or interrupt a running command. Press twice to fully exit.
`Up arrow`	Bring back your previous prompt. Edit and re-send.
`Cmd + K`	Clear the visible terminal output (your messages stay in context).

TIP · PLAN MODE IS YOUR FRIEND

For anything risky (touching multiple files, editing config, working in a real codebase), toggle plan mode first with `Shift + Tab`. Claude tells you what it's going to do before doing it. You approve, it acts. Saves a lot of "oh no" moments.

Stuck? Here's what's happening

Most "broken" Terminal moments are the same six problems. Pattern-match the symptom, run the fix.

THE CURSOR'S GONE, NOTHING TYPES

A command is still running. Either wait for it to finish, or press `Ctrl + C` to cancel it.

I'M LOST, NO IDEA WHERE I AM

Type `pwd` to see your current folder. Type `cd ~` to go straight home. Start over from there.

"PERMISSION DENIED"

The command needs admin rights. Some tutorials prefix with `sudo` to grant them. Don't run `sudo` blind. Read what the command does first. If you're not sure, ask Claude before running it.

"COMMAND NOT FOUND"

Either the tool isn't installed, or you typed it wrong. Check spelling first. If spelling's right, the tool needs installing (often via `brew install`).

CLAUDE CODE FROZE MID-RESPONSE

Press `Esc` to stop the response. If that doesn't work, `Ctrl + C`. If still nothing, close the Terminal window with `Cmd + W` and start fresh. Your work in files is safe, only the chat history is lost.

ACCIDENTALLY DELETED SOMETHING WITH RM

`rm` bypasses Trash. The file is gone unless you have Time Machine running. Check Time Machine before doing anything else. For Pack files, GitHub holds the last commit even if local is gone.

WHEN IN DOUBT · DON'T

If a command looks scary or you don't recognise it, don't run it. Paste it into a Claude session and ask "what does this do?" first. Two minutes of asking beats two hours of recovering.

THE UNIVERSAL ESCAPE HATCH

Cmd + W closes the Terminal window. Whatever's running stops. You can always open a fresh one. Treat Terminal windows as disposable, not precious.

The whole thing on one page

Bookmark this page. The rest is detail. This is the daily lookup.

Open / close

`Cmd + Space`	Spotlight, then type "terminal"
`Cmd + Q`	Quit Terminal
`Cmd + W`	Close window only
`exit`	End shell session

Where am I

`pwd`	Print current folder
`ls`	List files here
`ls -la`	List with hidden files
`cd ~`	Go to home folder
`cd ..`	Up one folder
`cd -`	Previous folder

Read files

`cat file`	Print whole file
`less file`	Page through, q to quit
`head file`	First 10 lines
`tail file`	Last 10 lines
`tail -f log`	Follow live updates

Move & copy

`cp a b`	Copy file
`cp -r f1 f2`	Copy folder
`mv a b`	Rename or move
`mkdir name`	Make new folder
`rm file`	Delete (no undo)

Search

`grep "x" file`	Find in one file
`grep -r "x" .`	Find across folder
`grep -i "x"`	Case-insensitive

Claude Code

`claude`	Start session here
`/help`	All commands
`/model`	Switch model
`/clear`	Reset chat
`/compact`	Free up context
`/quit`	End session
`Shift+Tab`	Toggle plan mode
`Esc`	Stop response

Pack patterns

`chmod 600 .env`	Lock secret file
`plutil -lint`	Check plist before reload
`launchctl unload`	Stop bot service
`launchctl load`	Start bot service

Stuck?

`Ctrl + C`	Cancel running command
`Esc`	Stop Claude response
`Cmd + W`	Nuke the window
`pwd`	Where am I
`cd ~`	Reset to home